

The surface code with a twist

Theodore J. Yoder¹ and Isaac H. Kim²

¹*Department of Physics, Massachusetts Institute of Technology*

²*IBM, Thomas J. Watson Research Center*

The surface code is one of the most successful approaches to topological quantum error-correction. It boasts the smallest known syndrome extraction circuits and correspondingly largest thresholds. Defect-based logical encodings of a new variety called twists have made it possible to implement the full Clifford group without state distillation. Here we investigate a patch-based encoding involving a modified twist. In our modified formulation, the resulting codes, called *triangle codes* for the shape of their planar layout, have only weight-four checks and relatively simple syndrome extraction circuits that maintain a high, near surface-code-level threshold. They also use 25% fewer physical qubits per logical qubit than the surface code. Moreover, benefiting from the twist, we can implement all Clifford gates by lattice surgery without the need for state distillation. By a surgical transformation to the surface code, we also develop a scheme of doing all Clifford gates on surface code patches in an atypical planar layout, though with less qubit efficiency than the triangle code. Finally, we remark that logical qubits encoded in triangle codes are naturally amenable to logical tomography, and the smallest triangle code can demonstrate high-pseudothreshold fault-tolerance to depolarizing noise using just 13 physical qubits.

I. INTRODUCTION

The surface code [1–3] is a dominating proposal for nearest-neighbor quantum error-correction in a plane. But there are some good reasons for its ubiquity. For instance, asymptotically in the code distance and up to a constant factor, the surface code uses the fewest number of qubits per logical qubit. Also, the surface code has optimally sized checks (weight-four) for a topological stabilizer code [4] and a simple scheme for syndrome extraction using only one ancilla qubit per check [5]. Moreover, the syndrome information can be processed efficiently [2, 6–8], resulting in the highest known topological fault-tolerance threshold [9]. The huge body of work on the surface code inspires optimism – perhaps we have found the “best” topological code [10].

Of course, this is a difficult claim to justify completely in any rigorous sense. There are a plethora of other topological coding strategies offering advantages and disadvantages. As one example, subsystem topological codes can perform syndrome measurement by measuring only operators that are less than weight four [11, 12]. As another example, color codes [13] can implement Clifford gates transversally [14, 15] and thus more efficiently than the surface code, even if they fail to achieve as large a threshold [16]. Strictly speaking there is not even just one best strategy for computing with the surface code. One option is to encode multiple qubits into one large region of surface code by using defects called holes [17]. Braiding the holes results in logical operations on the encoded qubits [3], and, using the technique of magic-state distillation [18, 19], can achieve universality [20] limited only by T -depth of the circuit [21]. Alternatively, computing with surface code “patches”, where each logical qubit is localized to a square grid of physical qubits, can be done with lattice surgery [16, 22] and comparatively few qubits. Still, both these strategies use state distillation for the Clifford phase gate, $S = \text{diag}(1, i)$.

Surface code computation has more recently undergone another reformulation with the introduction of a different type of defect, called a *twist* [23, 24]. A twist defect is a weight-five check embedded in the surface code lattice. Unlike a hole, a twist destroys the Calderbank-Shor-Steane (CSS) [25, 26] nature of the surface code, as the weight five check contains Pauli Y . Four (or three) twists in a surface code lattice with uniform boundary can encode one qubit [24]. Interestingly, this twist encoding appears advantaged over the hole encoding, as the full Clifford group can be fault-tolerantly implemented by local operations and without state distillation for the phase gate S [24]. Using both the hole and twist encodings together is also possible [27] and implements the same gate set similarly. So far, however, while the advantages of twist defects have been explored for multiple defects within the same large lattice, there has been little said about possible advantages for patch-based schemes of computation, which, due to lower qubit overhead, are more friendly to current experiments.

This is our focus in this work, the development of a planar patch layout that uses twist defects to achieve full Clifford group computation with local operations. Universality can then be achieved by injecting and distilling magic states for $T = \sqrt{S}$ gates. We find that a single twist defect placed in the middle of a patch of surface code suffices for this task. Moreover, the typical formulation of a twist as a weight-five check can be simplified in our case to weight-four. The family of non-CSS codes corresponding to these twist-containing patches we call *triangle codes*, because the physical qubits making up a single patch can be arranged to fit within a triangle in the plane. We also show how triangle codes are related by lattice surgery to the traditional surface code, and use this to deduce a surgical method for implementing S on patches of surface code in a nonstandard planar layout.

In addition to providing a full, distillation-free implementation of the Clifford group, triangle codes have fur-

ther advantages over patches of surface code. First, triangle codes of (odd) distance d use $3d^2/4 + 1/4$ data qubits per logical qubit, besting the (rotated [5]) surface code's d^2 . While this does not asymptotically beat some color codes [16], which are yet another constant factor better, triangle codes also benefit from near surface code thresholds. Indeed, depolarizing-noise fault-tolerant circuits for syndrome extraction on triangle codes can be made nearly as simple as that on surface codes, using the minimal number of timesteps when amortized over many rounds of extraction and nearly the minimal number of ancillas (i.e. one per stabilizer generator).

A second advantage of triangle codes is their symmetry – logical Pauli operators \bar{X} , \bar{Y} , and \bar{Z} can be taken to each lie along a different side of the triangular layout of qubits. This means that fault-tolerant measurement and initialization in any Pauli basis is possible, albeit not as straightforwardly as on CCS codes like the surface code. Additionally, triangle codes possess transversal order-3 single-qubit Clifford gates, which cyclically permute the Paulis, up to a permutation. For a single patch of triangle code, the fact that a permutation is required can be ignored, and a logical order-3 gate, plus logical initialization and measurement of the patch, suffice for tomography (or randomized benchmarking) of the logical qubit. Noting that for $d = 3$ this can be done with just 13 physical qubits including ancillas makes this a promising scheme for first implementations of complete quantum fault-tolerance.

In Sec. II we define the triangle codes, calculate their threshold for topological memory, and suggest circuits for syndrome extraction. In Sec. III, we discuss initialization and measurement of triangle code patches. The strategies we put forward there translate into schemes for computation with tessellating triangle patches in the plane, Sec. IV. In Sec. V we discuss circuits specifically for the smallest distance-3 triangle code, and calculate pseudothresholds. Sec. VI concludes.

II. CONSTRUCTING AND ERROR-CORRECTING TRIANGLE CODES

Our first task is to construct the family of triangle codes. The signature element of a triangle code is a central twist defect, and so to begin, we show in Fig. 1 how the triangle codes arise from the dislocation codes [23, 24] by lattice surgery. Notice that this surgery simplifies the code – all stabilizer generators are now at most weight-4 and fewer physical qubits are required. Further lattice surgery can symmetrize or extend the triangle code into the more general family that we describe now.

A general $r \times s \times t$ triangle code is described by three positive integer parameters r, s, t . We find the codes are most easily described by placing qubits at points in three dimensions, $(x, y, z) \in \mathbb{Z}_r \times \mathbb{Z}_s \times \mathbb{Z}_t$ for $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$, subject to the constraint that at least one of x, y, z is zero. In other words, this places

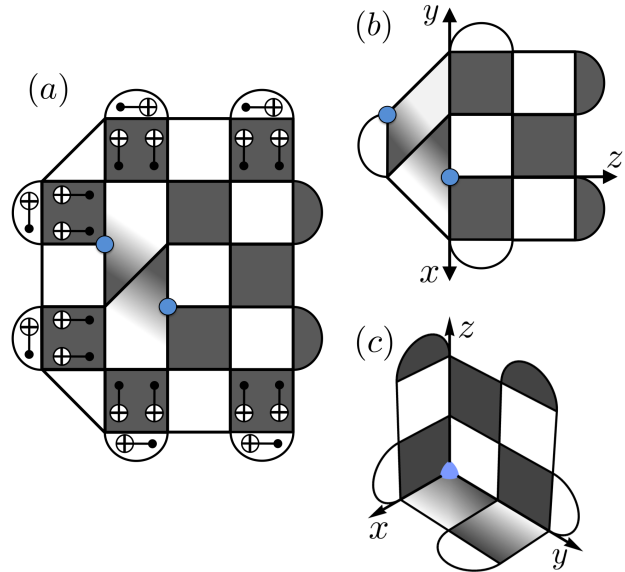


FIG. 1. Converting a dislocation code [23, 24] into an asymmetric triangle code. In (a) a dislocation code with two twists encodes one logical qubit (physical qubits are on the vertices or the lattice pictured). Stabilizers, both plaquettes and loops, are colored dark gray if they are X -type and white if they are Z -type. Stabilizers graded white to gray are mixed-type, and consist of X s where they are gray, Z s where they are white, and a Y on the blue, dotted qubits. Overlaid is a local Clifford circuit of CNOTs that takes the code to (b) when the inner CNOTs are applied, followed by the outer. Note that many qubits are unentangled from the bulk of the code during this process. From (b) to (c), a single-qubit Clifford is applied to remove a Pauli Y from one stabilizer, and the code is reoriented to get an asymmetric triangle code.

qubits on integer lattices in the xy -, yz -, and xz -planes as seen in Fig. 2(a). It should be noted that placing qubits in three dimensions is entirely for convenience, and a projection into two dimensions is readily obtained, Fig. 2(b). The projection fits within the eponymous triangle.

Stabilizers of a triangle code are local Pauli operators of either weight-4 (plaquettes) or weight-2 (loops). Plaquettes are associated with half-integer lattice points in the xy -, yz -, and xz -planes. They may be X -type (consisting of only Pauli X and I operators), Z -type (only Pauli Z and I), or mixed-type (any other combination of Paulis). By convention, we will choose the plaquette at $(1/2, 0, 1/2)$ to be X -type. Plaquettes, in order to commute, must alternate X -type and Z -type whenever possible (e.g. implying the plaquette at $(0, 1/2, 1/2)$ is Z -type), but geometry demands that mixed-type plaquettes are placed starting from the origin and extending in some direction to the outer edge. We conventionally take the plaquettes associated to the points $(1/2, m + 1/2, 0)$, for $m \in \mathbb{Z}_{s-1}$ to be mixed type. The central mixed-type plaquette at $(1/2, 1/2, 0)$ is special in that it contains Pauli Y on the qubit $(0, 0, 0)$, while all other mixed-type pla-

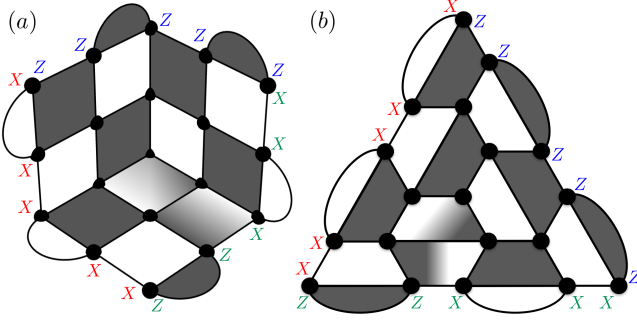


FIG. 2. (a) The distance-5 triangle code viewed as three conjoined surface codes in 3D. Black dots are data qubits, and stabilizers are colored as in Fig. 1 except for the Pauli Y on the origin that is part of the central mixed-type stabilizer, which is now left implicit. Along the sides, a string of Z s (blue) is \bar{Z} , a string of X s (red) \bar{X} , and a mixed-type string (green) \bar{Y} . (b) Shows the same, but in a plane.

quettes are half X and half Z .

Loops are placed with support on every other pair of qubits along the boundary. They can also be X -type, Z -type, or mixed-type. By convention, we will choose Z -type loops to be associated with pairs of qubits for which $x = r - 1$. This fixes the position and type of all other loops.

By counting qubits and stabilizer generators it can be checked that triangle codes for any r, s, t , encode one logical qubit. Logical operators \bar{X} , \bar{Y} , and \bar{Z} may be taken to lie on the boundary, crossing the x -, y -, and z -axes, respectively, as shown in Fig. 2(a). If $P_{(x,y,z)}$ denotes a Pauli acting on the qubit at (x, y, z) ,

$$\bar{X} = \prod_{j \in \mathbb{Z}_s \setminus 0} X_{(r-1,j,0)} \prod_{k \in \mathbb{Z}_t} X_{(r-1,0,k)}, \quad (1)$$

$$\bar{Y} = \prod_{i \in \mathbb{Z}_r \setminus 0} Z_{(i,s-1,0)} \prod_{k \in \mathbb{Z}_t} X_{(0,s-1,k)}, \quad (2)$$

$$\bar{Z} = \prod_{i \in \mathbb{Z}_r \setminus 0} Z_{(i,0,t-1)} \prod_{j \in \mathbb{Z}_s} Z_{(0,j,t-1)}. \quad (3)$$

The minimum weights of the logical operators are respectively, $d_x = s + t - 1$, $d_y = r + t - 1$, and $d_z = r + s - 1$. We describe symmetric triangle codes that have parameters $r = s = t = (d + 1)/2$ by the adjective distance- d .

Triangle codes may also be more directly constructed from a surface code (and vice versa). This is not entirely surprising when the corners of the surface code are viewed as twists [27]. A fault-tolerant method to perform this conversion is shown in Fig. 3. Indeed, in our notation and with our conventions, a $d \times d$ surface code is precisely a $d \times 1 \times d$ triangle code.

So far we have described triangle codes as topological codes, but we have not yet proved that a threshold for storing quantum information exists. In Appendix A, we show such a threshold exists through a straightforward argument analogous to that for the toric code [2]. For the

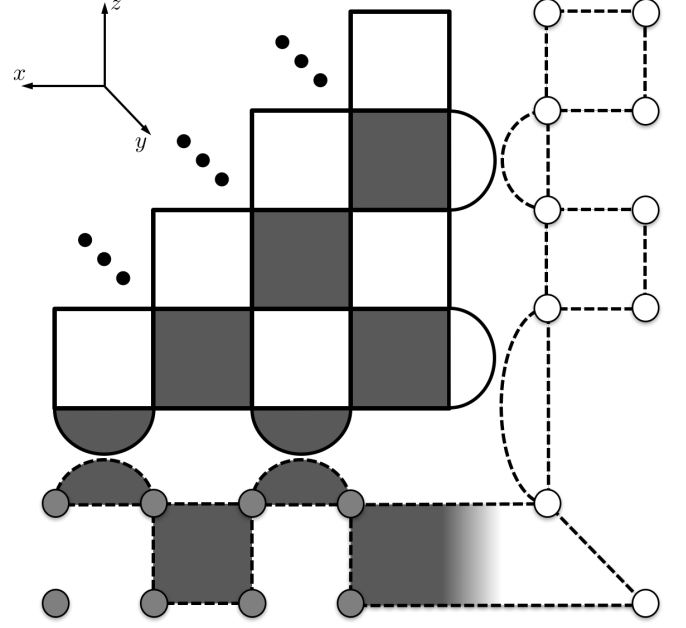


FIG. 3. Creating a triangle code from a $d \times d$ surface code by extending from a corner. In the diagram, ancilla qubits are prepared in $|+\rangle$ (gray) or $|0\rangle$ (white). Then, the stabilizers of the triangle code are measured $O(d)$ times. The code is always in the $+1$ -eigenstate of the dashed plaquettes and the product of dashed loops and solid loops that together make up a plaquette of the triangle code. Thus, syndromes from half of the triangle code stabilizers in the xy - and yz -planes can be used reliably for decoding. The effective distance of this procedure is d .

| code | syndrome | X threshold | X-Z threshold |
|----------|----------|-----------------|--------------------------|
| triangle | ideal | $\approx 10\%$ | $\approx 10\%$ |
| rotated | ideal | $\approx 10\%$ | $\approx 10\%$ |
| triangle | noisy | $\approx 3.2\%$ | $\approx \mathbf{2.6\%}$ |
| rotated | noisy | $\approx 3.2\%$ | $\approx \mathbf{3.2\%}$ |

TABLE I. Numerical threshold estimates for the triangle and rotated surface codes show appreciable differences only for noisy syndrome measurements and bit-phase flip noise (bold).

purpose of comparison with the surface code, we also estimate thresholds for the triangle code and rotated surface code [5] families under phenomenological noise. We consider both bit (X) and bit-phase (X - Z) flip storage errors and measurement models where the measured syndromes are ideal or noisy. Table I summarizes the results, and Appendix A provides more detail on the simulation.

We now turn to syndrome extraction circuits for the triangle codes. The key notion here is that of effective distance. We say that the effective distance of a syndrome extraction circuit is d if no fewer than d faulty circuit components (e.g. single-qubit gates, two-qubit gates, preparations, measurements) can create a logical error on the data, while also triggering no syndrome measurements. This implies being able to correct any set of $\lfloor d/2 \rfloor$ faults given the syndrome, because no two such sets can

have the same syndrome. We will assume the depolarizing noise model for faults when designing these circuits.

A $d \times d$ surface code possesses a very simple method [5] for syndrome extraction with effective distance d under depolarizing noise, using an optimal number of ancilla qubits – one per stabilizer check – and an optimal number of timesteps – six, if we say coupling the ancilla to all four data qubits takes four total timesteps and initialization and measurement each take another. This procedure can leave two-qubit errors on the data from a single fault (commonly called a “hook” error [2]) but such errors are oriented such that d hook errors are required to write a logical error onto the data (see Appendix B).

We might hope that the triangle code, being for the most part surface code, also supports as simple a syndrome extraction. This is nearly the case. By brute-force check of the distance-3 symmetric triangle code, we actually find no fault-tolerant procedure using both the minimal number of qubits and the minimal number of timesteps. This suggests we have to think a little more creatively to find a simple syndrome extraction circuit for the triangle codes. Luckily, we do not have to use too many more resources. Two qubits in addition to the allotted one per stabilizer check suffice to perform full-distance extraction on the triangle code. In fact, amortized over many rounds of syndrome extraction, our protocol also uses the minimal number of timesteps per round, six.

Our syndrome extraction circuit is shown in Fig. 4. The design uses the standard scheduling of the surface code [5] within a plane, and staggers the timing of different planes to avoid conflicts in coupling to the data qubits. To achieve fault-tolerance to the maximum number of depolarizing faults, even this is not enough, however. For example, just two hooks can cause the weight three error $Z_{(1,0,0)}Z_{(0,0,0)}Z_{(0,1,0)}$ and the remaining $d-3$ Paulis making up \bar{Z} (considered equivalently with support across the x - and y -axes) can be caused by single qubit faults. Thus, \bar{Z} may be written onto the data with only $d-1$ faulty circuit components. On the other hand, reducing the distance by one is actually the worst that can happen in this design (see Appendix B). A larger triangle code using $3(d+1)^2/4 + 1/4$ qubits (for even d), still with asymptotic qubit count of $3d^2/4$, will perform at effective distance d even when syndromes are extracted using just one ancilla per check.

If we wish to make syndrome extraction for a distance- d triangle code with effective distance d , we can detect hook errors by creating and decoding 2-CATs (see Fig. 5) for (any) two of the plaquettes adjacent to the origin. This efficiently deals with the case noted above, where a pair of hook errors near the origin can write three errors onto the data, and is represented in Fig. 4 as a pair of ancillas in two of the central plaquettes.

Neither syndrome extraction with 2-CATs nor extraction on a larger code are ideal for distance three, if the goal is to make the most qubit-efficient fault-tolerant architecture with high pseudothreshold. In Sec V, we cre-

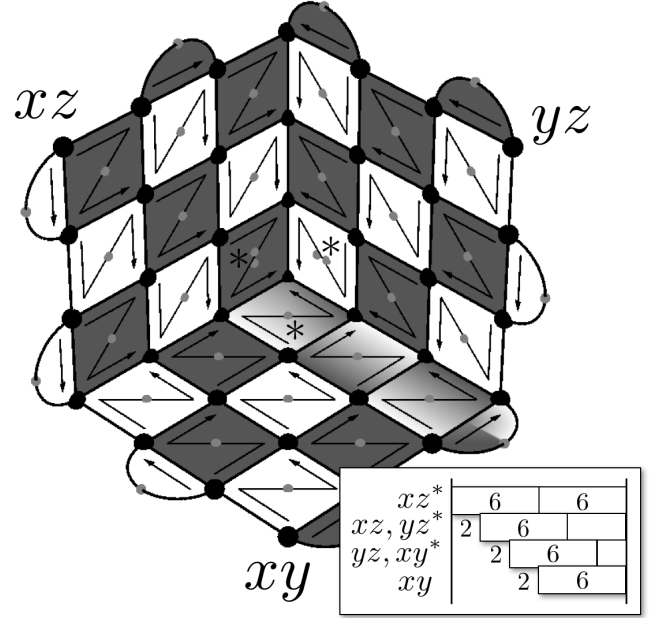


FIG. 4. A schematic of the syndrome measurement on the triangle code. The arrows show the order (tail to head) in which an ancilla couples to the data after it is initialized. Two plaquettes adjacent to the pivot are endowed with flag qubits to detect hook errors, as described in Fig. 5. The inset shows that syndrome extraction can be fit together so, over many cycles, the amortized time per complete syndrome extraction is six timesteps. The three planes (labeled xz , yz , xy) are measured two timesteps offset and starred (*) stabilizers, the plaquettes adjacent to the origin, begin their measurement cycle two timesteps earlier than the rest of their plane to avoid time conflicts when coupling to the data.

ate a syndrome extraction circuit specialized to distance three using one ancilla per check.

III. INITIALIZATION AND MEASUREMENT

The fact that triangle codes are not CSS complicates their initialization and measurement protocols. This is actually the biggest challenge to computing with them. Gates, in contrast, which are discussed in detail in Sec. IV, can be performed with a combination of known lattice surgery techniques [22] supplemented with 1-bit teleportation [28] and, for non-Cliffords, magic-state injection and distillation [18].

As an example of the complications that arise in measuring a non-CSS code, consider attempting to deduce the eigenvalue of \bar{Z} by measuring all qubits of a symmetric, distance- d triangle code in the Z -basis. The parity of the qubits across the top of the code should equal \bar{Z} . But can we error-correct this value with maximum distance? In fact, we cannot. Because we measured all qubits in the Z -basis, we are lacking the information from the central mixed-type stabilizer. Thus, a string of X errors

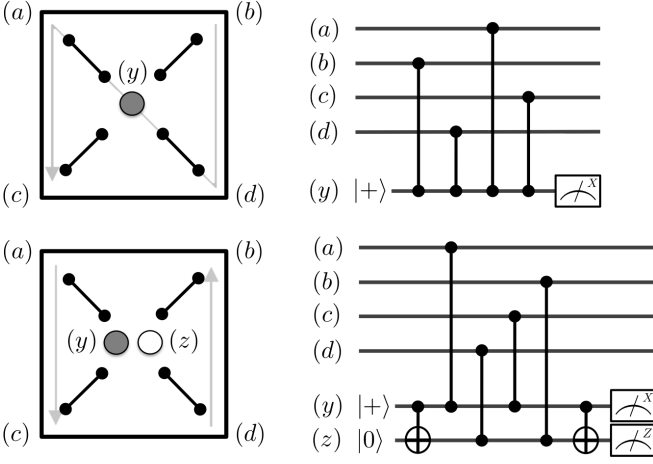


FIG. 5. The conventional single-qubit syndrome measurement (top row) and the flagged syndrome measurement (bottom row) both take six timesteps (including preparation and measurement). The gates $\bullet\text{---}\bullet$ in this example of measuring a Z -type plaquette are controlled- Z . In Fig. 4, we suggest using the flag qubits on two plaquettes near the pivot to achieve full-distance syndrome extraction.

from the origin to the top of the code is an undetectable weight $(d+1)/2 < d$ error that causes our measurement to fail. In fact, any measurement scheme on only individual qubits also cannot achieve maximum distance. We can either be satisfied with this noisy measurement (perhaps in an implementation with extremely reliable single-qubit measurements), or develop full-distance alternatives.

We present three different approaches that achieve full-distance initialization and measurement. Each approach results in a different scheme in Sec. IV for computing with the triangle codes. We feel each has something to offer and we discuss tradeoffs later. In Sec. V, we focus on the distance-3 triangle code, for which we have developed specialized alternatives to these schemes.

The first method we call code conversion, simply because it works by using Fig. 3 to convert between the surface code, for which initialization and measurement procedures are simple and known, and the triangle code. This actually moves any logical state between the surface code and the triangle code. Notice, however, that this is stronger than necessary when all we want is to initialize or measure a Pauli eigenstate. For this reason, the second method, which we call basis-state conversion, uses fewer qubits to convert a Pauli basis state from the surface to triangle codes or measure the basis-state transversally. The final method, employing CAT states, is the most qubit efficient but least time efficient.

To understand the first approach, notice that a distance $2d + 1$ triangle code admits transversal measurement of \bar{X} , \bar{Y} , and \bar{Z} with effective distance d by simply measuring all qubits in the X -, Y -, or Z -bases. Initialization of a state in the distance $2d + 1$ triangle code can also be done by initializing a $d \times d$ surface code patch (for which there are known procedures [5, 10] with effective

distance d) and then extending the patch using Fig. 3. This is the “code conversion” technique for initialization and measurement. This procedure is very much based on the surface code, and so storing data with distance $\geq d$ could be done without ever converting to the triangle code at all. However, the embedding of the surface code patch in a large triangular patch is important for performing single-qubit Clifford gates, as explained in Section IV.

In “basis-state conversion” we also prepare a basis-state, say $|\bar{0}\rangle$, in a $d \times d$ surface code, but instead of treating this surface code as $1/3$ of a triangular patch, we treat it as $2/3$, the xz - and yz -planes say. Conversion to the surface code can then be done by ceasing measurement of the loop operators along the bottom of the surface code, and measuring the stabilizers of a $(d+1)/2 \times (d+1)/2 \times d$ triangle code $O(d)$ times (see Fig. 6). Regardless of the initial state of the new qubits, this process has effective distance d , since \bar{X} must always cross the original $d \times d$ surface code patch, whose Z -type stabilizers are always reliable throughout the conversion. A \bar{Z} error is ignorable since we are preparing $|\bar{0}\rangle$. After this conversion, symmetrize the triangle code, so that we are left with the symmetric, distance- d code, by measuring its stabilizers and ceasing to measure the extraneous stabilizers in the topmost rows. Again, \bar{Z} can occur from fewer than d faults but is irrelevant. Logical measurement is done by inverting the last step — extend the triangle code along the z -axis $d/2$ steps, and measure all qubits in the Z -basis.

The CAT-state method to initialization and measurement is probably the most straightforward. Using a row of d ancilla qubits along an edge of a distance- d triangle code, we can create and verify a d -CAT state for measuring the logical operator also lying along that edge. Creating the d -CAT with local operations takes d timesteps (e.g. by measuring two-bit parity checks starting from $|+\rangle^{\otimes d}$ [29]), and during this process, we must be collecting syndromes from the patch. We must also repeat the CAT-state measurement $O(d)$ times to reliably measure the logical state, leading to $O(d^2)$ time for logical measurement. Initialization is essentially the same process, using $O(d^2)$ time to measure a logical operator $O(d)$ times and the stabilizers $O(d^2)$ times to project the code onto the desired logical state.

IV. PLANAR, NEAREST-NEIGHBOR COMPUTATION WITH TRIANGLES

With initialization and measurement protocols created, our next priority is to perform the Clifford group on triangle codes. Inevitably, by desiring to implement the two-qubit gate CNOT, we are forced to consider the layout of several triangle codes within the same planar geometry. The basic design is to tile the plane with equilateral triangles, each having the potential to encode a qubit as per Fig. 2(b). However, the size of these code

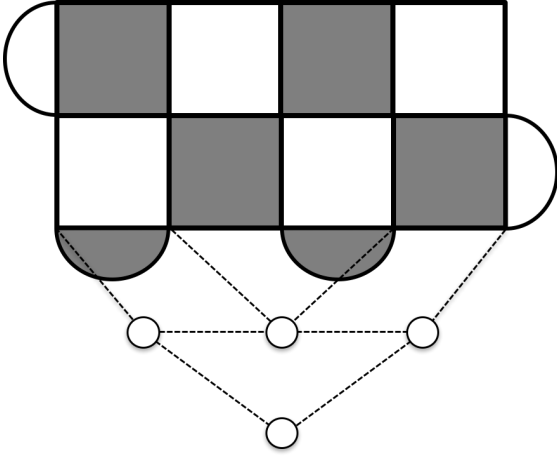


FIG. 6. Converting a $d \times d$ surface code (in this case $d = 5$ with the top two rows of qubits omitted) to a $(d+1)/2 \times (d+1)/2$ triangle code. To do so, stop measuring the bottom loops (here the two X -type ones) and start measuring the stabilizers of the triangle code (dashed skeleton) for $O(d)$ cycles. The initial state of the new qubits is not relevant to the effective distance, but one might be able to correct more higher weight errors if they were initialized as, for instance, a collection of 2-qubit CAT states or as a $(d-1)/2 \times (d-1)/2$ surface code.

| init. & meas. scheme | qubits/log. | 1-qubit gate | CNOT |
|------------------------|-----------------|--------------|----------|
| code conversion | $3d^2 + O(d)$ | $O(d)$ | $O(d)$ |
| basis-state conversion | $9d^2/4 + O(d)$ | $O(1)$ | $O(d)$ |
| CAT states | $6d^2/4 + O(d)$ | $O(1)$ | $O(d^2)$ |

TABLE II. A comparison of three strategies for planar Clifford computation in triangle code geometry with distance d . Compared quantities are qubit counts per logical qubit and Clifford gate times.

patches, and also which patches encode computational data and which must act as ancillas used for gates, varies depending on the initialization and measurement protocol used from Sec. III. So, we present three designs that vary in these ways, and thus also vary in how space efficiently data is stored and how time efficiently gates are performed on that data (see Table II and Fig. 7).

In Fig. 7(a), we show the first of our three designs corresponding to performing initialization and measurement with the surface code. Indeed, in this design we actually prefer to use the surface code (1/3 of a triangular patch) as our resting code, holding the data between gates. Standard lattice surgery [22] suffices for performing CNOT gates between the surface code patches. However, the large triangular patches become important when we wish to perform single-qubit Clifford gates.

The idea is shown in Fig. 8. Through a composition of 1-bit teleportation (Fig. 9) and code conversion (Fig. 3), we can place \bar{Y} on the edge of the surface code. Enforcing the relabeling $\bar{Y} \rightarrow \bar{X}$, while keeping \bar{Z} fixed performs the \bar{S}^\dagger gate. If we instead performed 1-bit teleportation

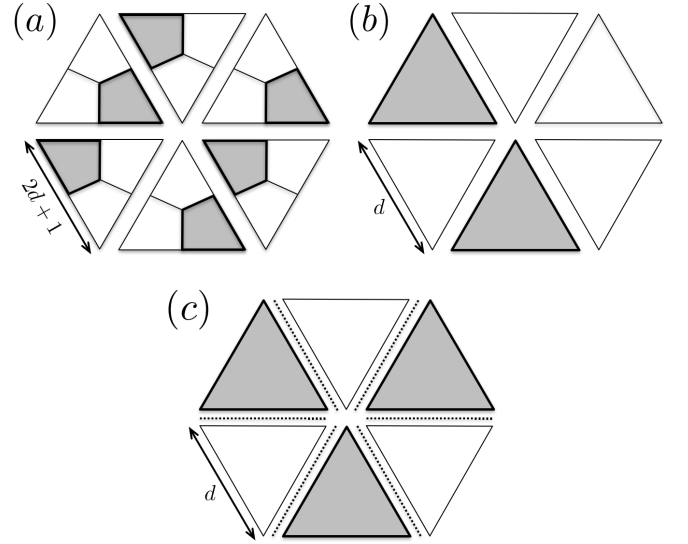


FIG. 7. Tessellating patches of triangle code capable of complete Clifford computation with effective distance d , corresponding to using (a) code conversion (b) basis-state conversion and (c) CAT states for initialization and measurement. In a resting configuration, the shaded regions represent patches of physical qubits that are encoding data, while blank areas represent ancillary patches. In (a) the resting code is actually the surface code, and the triangular patches (with qubit layout as in Fig. 2(b)) are present only to facilitate single-qubit gates. In (b), we use two ancilla patches per data patch to perform gates, because initialization and measurement use 2/3 of a neighboring patch. In (c), we add ancillary qubits between the patches to make CAT states for initialization and measurement, and so only require one ancillary patch per data patch to achieve arbitrary reorientation and CNOTs with neighboring data patches.

into the top sector of Fig. 8(b), we can perform the gate $\bar{H}\bar{S}^\dagger\bar{H}$, completing a set of single-qubit Cliffords. We can also get \bar{H} directly by the composition of three 1-bit teleportations to move $|\psi\rangle$ around the twist (in either direction). The fact that we can perform the whole Clifford group on surface codes within a triangular patch layout is quite relevant to surface code research. Indeed, besides state distillation, there is only one other method we are aware of for performing S on planar surface code patches [30], and it effectively involves conversion to the color code through folding [31]. It seems the triangle code offers a more natural approach, albeit with a not insignificant cost of $3d^2 + O(d)$ physical qubits per logical.

Lower overhead computation can be achieved by using the triangle code as the resting code. Our second design Fig. 7(b) uses the basis-state conversion scheme of initialization and measurement. In this case, whenever we want to measure or initialize a patch we require ancilla qubits extending outside of the triangular region of the patch (see III). Thus, we require a neighboring ancilla patch empty of data. Moreover, the ancilla must be adjacent to the side containing the logical Pauli we want to measure or initialize an eigenstate of. This influences

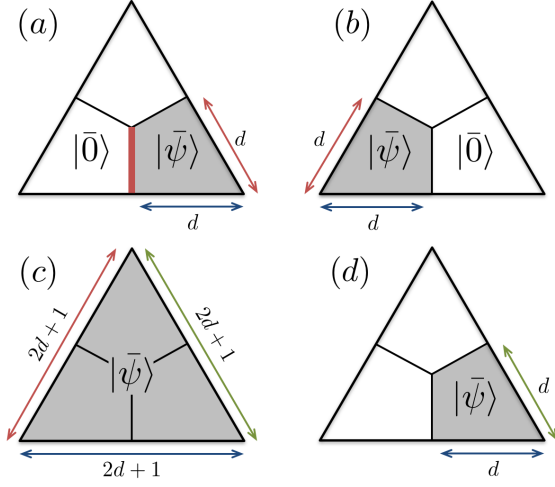


FIG. 8. Performing S^\dagger on a surface code patch embedded in $1/3$ of a triangular patch as in Fig. 7. Colored arrows represent the directions of logical Paulis \bar{X} (red), logical \bar{Z} (blue), and logical \bar{Y} (green). In (a), we prepare an ancilla in $|\bar{0}\rangle$ and project onto the $+1$ -eigenspace of $\bar{X}_A \bar{X}_D$ of the ancilla and the data (thick, red line). In (b), we project what was the original data sector to $|\bar{0}\rangle$, completing the 1-bit teleportation (see Fig. 9) of $|\bar{\psi}\rangle$ to the ancillary sector. In (c), we extend the surface code to the full triangle code of distance $2d+1$, using Fig. 3, then, in (d), shrink the triangle code back down to the original sector. In the end, \bar{Y} is located on the edge which originally held \bar{X} . Thus, $\bar{Y} \rightarrow \bar{X}$ and $\bar{Z} \rightarrow \bar{Z}$, so we have implemented \bar{S}^\dagger .

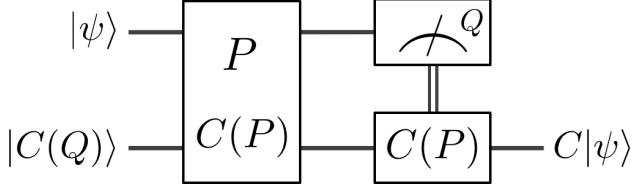


FIG. 9. A family of circuits for teleporting Clifford gates. One may choose any two (different) single-qubit Paulis P, Q and any single-qubit Clifford C (including identity). Then, let $C(R) = CRC^\dagger$ and also let $|R\rangle$ be the $+1$ eigenstate of single-qubit Pauli R . In the diagram, the first box represents a projector onto the $+1$ eigenspace of $P \otimes C(P)$.

all of our gate designs that use specially prepared ancilla patches.

For instance, a projector circuit for CNOT is shown in Fig. 10. To use this diagram with basis-state conversion initialization and measurement necessitates two ancilla patches – (1) an ancilla patch adjacent to both control and target and (2) a second ancilla patch adjacent to the first – and, moreover, that the data patches serving as control and target have their \bar{Z} and \bar{X} sides adjacent to the first ancilla, respectively. The layout of Fig. 7(b) guarantees the availability of the two ancillas. To ensure that the control and target patches are correctly oriented, notice that reorienting data patches can be done using 1-

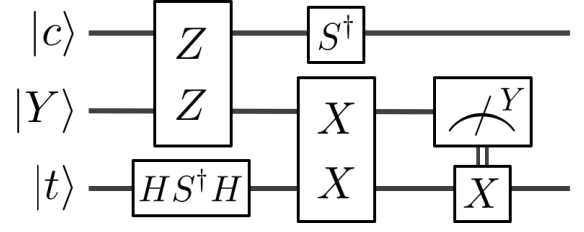


FIG. 10. A circuit performing CNOT from $|c\rangle$ to $|t\rangle$ in the spirit of [22]. The ZZ and XX boxes are projectors onto the $+1$ eigenspace of those operators. We use this design specifically for computation in the layout of Fig. 7(b), where preparing and measuring the ancilla in the Y -basis is most natural, because that is the side of the ancilla patch that is not adjacent to data.

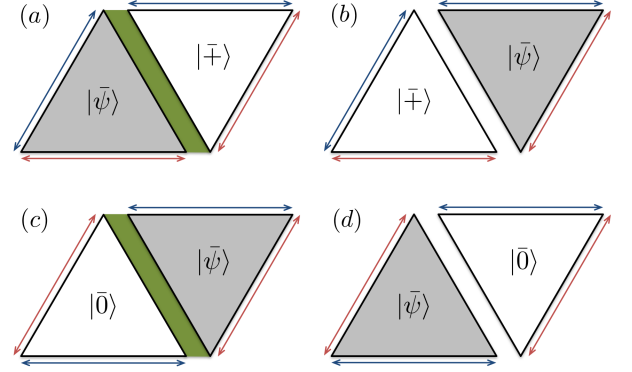


FIG. 11. Reorienting a triangular patch by a reflection using a composition of two 1-bit teleportations Fig. 9. Colored arrows indicate the \bar{X} (red) and \bar{Z} (blue) sides of a patch, and green parallelograms indicate projection onto $\bar{Y}\bar{Y}$. Notice that going from (b) to (c) is simply a relabeling of the ancilla patch — $|\bar{\pm}\rangle$ oriented as in (b) is the same state as $|\bar{0}\rangle$ oriented as in (c). It is possible to view this process as a non-reorienting Hadamard (implemented by 1-bit teleportation with $C = H$ in Fig. 9) followed by a relabeling Hadamard (choosing to treat \bar{X} as \bar{Z} and vice versa), a composition which implements logical identity but changes orientation.

bit teleportation in the procedure prescribed by Fig. 11. When Fig. 7(b) is tessellated, there is just enough room to reorient any data patch however we like, though not necessarily in parallel with some of the nearest other data patches. Both the reorientation and the CNOT itself take $O(d)$ time.

Single-qubit Clifford gates on the Fig. 7(b) layout can be done by simply relabeling sides of the code, treating, for instance, what was the \bar{Z} side as \bar{X} and vice versa. This method of single-qubit gates directly exploits the symmetry of the triangle code — whatever subsequent lattice surgery might be done with one side can equally well be done with another. For this to work, though, it is crucial that we have the aforementioned reorienting protocol.

Our final layout is shown in Fig. 7(c) and uses d ancilla qubits positioned between triangles. In this case,

logical initialization and measurement can be done without intruding upon neighboring patches, making gates quite simple conceptually. Fig. 10 suffices for coupling neighboring patches, Fig. 11 for reorienting them, and relabeling sides for single-qubit Clifford gates. The ancilla qubits do not get in the way of lattice surgery since we can always extend a triangular patch through them. The downside is the $O(d^2)$ time it takes to initialize and measure ancillas for a CNOT gate (and for reorientation).

V. THE SMALLEST TRIANGLE CODE, CODE COMPARISON, AND LOGICAL TOMOGRAPHY

Our goal in this section is to develop circuits for syndrome extraction, initialization, measurement, and gates on the smallest, fully fault-tolerant member of the triangle code family.¹ We view this as a worthy pursuit because of the qubit savings over the smallest, fully fault-tolerant surface code – 13 versus 17. We will also see that the triangle code provides other advantages, such as the ability to perform tomography on the encoded logical qubit, also with only 13 qubits. We calculate pseudothresholds for our circuits, and find them comparable to the surface code, and better than other small fault-tolerant designs such as the color code. Thus, even though we do not claim our circuits are optimal in terms of depth or pseudothreshold, they at least show by construction that the smallest triangle code has the potential for a small demonstration of fault-tolerance.

Our first task is to create a circuit for syndrome extraction on the distance-3 triangle code. As mentioned in Sec. II, we can check by enumeration that this is impossible if we make the harshest demands on space and time requirements — just 6 ancillas and 6 timesteps are not sufficient for error-correction on the distance-3 triangle code. Alternatively, the design of Fig. 4 implies 8 ancillas (a total of 15 qubits) and 7 timesteps are sufficient to perform syndrome extraction for distance three. We now discuss a circuit identity that we can use to reduce those 8 ancillas to 6 (a total of 13 qubits), but with 8 timesteps.

The idea is illustrated in Fig. 12, where extraction of a loop’s syndrome is *interwoven* with extraction of the neighboring plaquette’s syndrome. This ordering necessitates a CZ gate between the loop and plaquette ancillas before measuring, but that same CZ gate also endows the loop ancilla with the ability to detect hook errors caused by failure of the plaquette ancilla. Ideally, this CZ gate can be done directly between the ancillas, though this demands high connectivity (degree-5) for the plaquette ancilla. If degree-4 connectivity is desired, a depth-2 “cascade” of CNOT and CZ gates can be used to perform

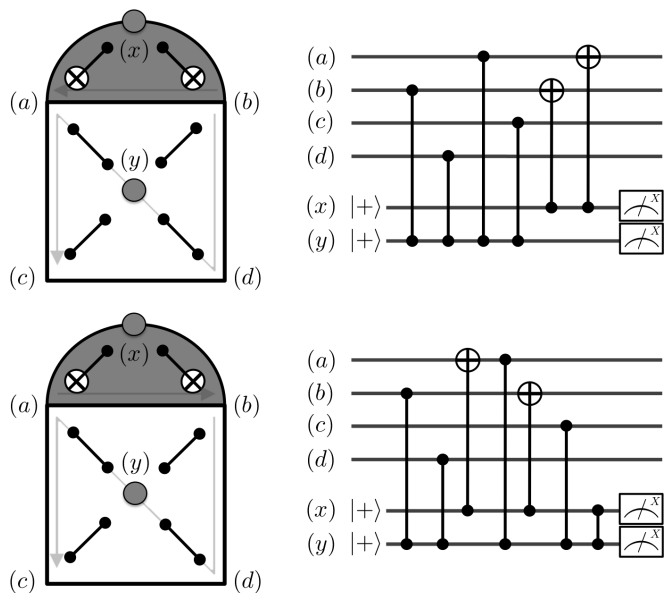


FIG. 12. A plaquette stabilizer and loop undergoing syndrome extraction. The top image and circuit shows the conventional ordering, where coupling to the loop occurs completely after coupling to the plaquette. The lower circuit is equivalent (by circuit identities), but here coupling to the loop is interwoven with coupling to the plaquette. This rewriting, however, makes a significant difference to fault-tolerance, because the lower circuit can detect hook errors (e.g. an X occurring partway on qubit (x) will propagate a Z onto qubit (y) flipping its measurement) while the upper cannot.

the CZ gate between ancillas using a data qubit as an intermediary. This is still fault-tolerant (data qubits are not coupled to one another), but will lower the threshold due to larger circuit depth.

Using interwoven extraction on two plaquettes instead of 2-CATs provides a circuit for distance-3 syndrome extraction with only 6 ancillas, Fig. 13. For larger distance triangle codes, we leave the question open as to whether a similarly interwoven circuit suffices for full-distance extraction. In any case, in the large distance limit, a two-qubit savings over Fig. 4 is less impactful.

In Tab. III, we show a comparison of small distance-3 fault-tolerant designs by overhead and pseudothreshold for syndrome extraction. These designs were chosen as they satisfy reasonable criteria for a near-term fault-tolerant qubit: all use less than 20 qubits including ancillas, the connectivity between qubits required is low, and no design uses postselection to create ancillas during syndrome extraction. The triangle code performs remarkably well in the comparison, saving two or four qubits over the surface code with a drop in pseudothreshold of only 3% or 14%, respectively. In Appendix C, we provide more detail on the circuits and the exact-counting procedure used to determine these pseudothresholds.

It is also important that we be able to initialize and measure logical states in the distance-3 triangle code. It turns out that each of the triangle code designs from

¹ The 7-qubit, distance-3 triangle code was actually the first member of the family discovered by Andrew Cross at IBM T.J. Watson through enumeration of 7-qubit stabilizer codes [32].

| Code | Data | Anc. | Qubits (#/surf.) | Max deg. | Lower pseudoth. (#/surf.) | Upper pseudoth. (#/surf.) |
|------------|------|------|------------------|----------|-----------------------------|-----------------------------|
| 5-qubit | 5 | 3 | 8 (.47) | 3 | 1.02×10^{-5} (.05) | 1.05×10^{-5} (.04) |
| 5-qubit | 5 | 6 | 11 (.65) | 3 | 2.55×10^{-5} (.14) | 2.69×10^{-5} (.10) |
| Color | 7 | 6 | 13 (.76) | 3 | 3.47×10^{-5} (.19) | 3.77×10^{-5} (.15) |
| 5-qubit* | 5 | 12 | 17 (1.0) | 4 | 5.20×10^{-5} (.28) | 5.70×10^{-5} (.23) |
| Color* | 7 | 12 | 19 (1.1) | 6 | 5.80×10^{-5} (.31) | 6.65×10^{-5} (.26) |
| Surface | 9 | 4 | 13 (.76) | 4 | 7.85×10^{-5} (.43) | 9.40×10^{-5} (.38) |
| Bacon-Shor | 9 | 8 | 17 (1.0) | 3 | 8.98×10^{-5} (.49) | 1.07×10^{-4} (.43) |
| Triangle | 7 | 6 | 13 (.76) | 4 | 1.05×10^{-4} (.57) | 1.22×10^{-4} (.49) |
| Triangle | 7 | 6 | 13 (.76) | 5 | 1.57×10^{-4} (.86) | 1.92×10^{-4} (.77) |
| Triangle | 7 | 8 | 15 (.88) | 4 | 1.76×10^{-4} (.97) | 2.23×10^{-4} (.90) |
| Surface | 9 | 8 | 17 (1.0) | 4 | 1.82×10^{-4} (1.0) | 2.47×10^{-4} (1.0) |

TABLE III. A comparison of small, fully fault-tolerant designs across qubit count, maximum degree connectivity required, and pseudothreshold (under depolarizing noise) for the exREC of a transversal single-qubit gate (e.g. identity). In parentheses, quantities are written as a fraction of the corresponding surface code quantity. The pseudothreshold upper bounds apply only to the particular circuits and error model considered, but are useful insofar as they can definitively prove separations between the logical error rates of different designs. Starred (*) designs use connectivity that is not planar. All designs are capable of logical qubit tomography with just the qubit resources listed, except for the Bacon-Shor and surface code designs, which are lacking initialization and measurement in the Y -basis. Although the surface code does possess the largest provable pseudothreshold in our study, it does not do so by much, and a fewer-qubit triangle code design may prove to be a more practical small logical qubit.

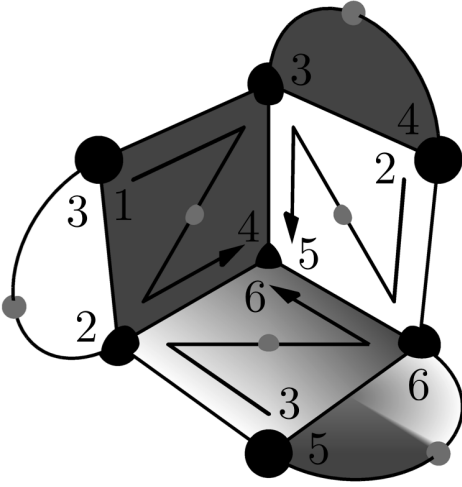


FIG. 13. Proposed syndrome extraction for the distance-3 triangle code using the minimal number of ancillas. Numbers indicate the timesteps in which coupling the ancilla to data occurs. Extraction from the X -type and Z -type plaquettes is interwoven with extraction on their neighboring loops. If the CZ gate between interwoven ancillas can be done in one timestep, then this whole circuit takes 8 timesteps (including preparation and measurement of ancillas) to extract the complete syndrome once. However, two subsequent extractions can be performed in 15 timesteps.

Tab. III can perform initialization and measurement using only the qubits provided. The idea is quite simple: to measure \bar{Z} we need only use the mixed-type plaquette

ancilla, because $\bar{Z} = Z_{(1,0,0)}Z_{(0,0,0)}Z_{(0,1,0)}$ is supported on qubits adjacent to it. Performing a measurement of \bar{Z} before a round of syndrome extraction, and repeating the process at most three times, will successfully measure \bar{Z} . Since \bar{X} and \bar{Y} are also localized around single plaquettes, similar procedures work for measuring them.

We show pseudothresholds for measurement of distance-3 triangle code designs in Tab. IV. They are lower than the pseudothresholds of transversal gates shown in Tab. III. This is to be expected from the slightly larger circuit depth, caused by having to measure \bar{Z} . Although a lower measurement threshold is not disastrous (measurement occurs at most once in a single-qubit experiment), we also show that, if necessary, the pseudothreshold can be raised by adding one additional qubit. Details are provided at the end of Appendix C.

The most reliable way of initializing the triangle code is through a non-fault-tolerant circuit followed by postselection. In Fig. 14, we show a depth-2 circuit that can prepare $|\bar{0}\rangle$ non-fault-tolerantly. Similar circuits exist for preparing eigenstates of \bar{X} and \bar{Y} . To guarantee fault-tolerance, Fig. 14 should be followed by postselection on trivial measurements of the syndrome and of \bar{Z} . Since this postselection is done only at the beginning of an experiment, we view it as not too restrictive for a small fault-tolerant design.

The logical operations we have described here, preparation and measurement in any Pauli basis and error-correction, serve to implement essentially any interesting single-logical-qubit experiment on the distance-3 triangle code. For instance, process tomography of logical

| Qubits | Max. Deg. | Lower | Upper |
|--------|-----------|-----------------------|-----------------------|
| 13 | 4 | 9.84×10^{-5} | 1.18×10^{-4} |
| 15 | 4 | 1.07×10^{-4} | 1.27×10^{-4} |
| 13 | 5 | 1.15×10^{-4} | 1.39×10^{-4} |
| 14 | 5 | 1.84×10^{-4} | 2.65×10^{-4} |

TABLE IV. A comparison of pseudothresholds for measurement exRECs of $d = 3$ triangle code designs. The first three designs match the three from Tab. III, and have lower pseudothresholds for measurement than they do for gates. The last design is the 13-qubit, degree-5 design augmented by an extra qubit that is involved in measuring the logical operator (see Appendix C, Fig. 23).

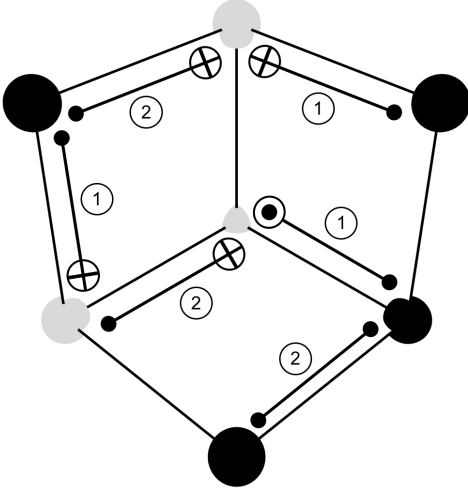


FIG. 14. Creating the $|\bar{0}\rangle$ state in the distance-3 triangle code can be done with this depth-2 circuit. Light qubits should be prepared in $|0\rangle$ and dark qubits in $|+\rangle$. Numbered buttons indicate the timesteps in which gates should be performed. The notation $\bullet\text{---}\bigcirc$ indicates a controlled- Y gate. Although the gates are pictured acting between data qubits, if the connectivity of the implementation does not cooperate, they can also be indirectly implemented by CNOT cascades through the ancilla qubits (not shown).

identity \bar{I} can be performed by preparing logical Pauli eigenstates and then performing logical measurement in a Pauli basis. Tomography of \bar{I}^k is the same with more rounds of error-correction between preparation and measurement. Single-qubit logical Cliffords are essentially rotations or reflections of the triangle, which can be done without need for physical Cliffords. If desired for tomographic purposes, physical Cliffords can be applied to transform to a locally equivalent triangle code without significantly affecting the circuits for syndrome extraction or measurement. A special case of this is a transversal, order-3 Clifford gate $\bar{S}\bar{H}$: perform H on all qubits at positions $x > 0$ or $x = 0, z > 0$ and $\bar{S}\bar{H}$ on the central qubit. The resulting code is the same as the initial

triangle code rotated 120° counterclockwise. Since the logical Pauli operators lie along the triangle's sides, they are cyclically permuted by this operation.

VI. CONCLUSION

It is important to keep in mind that, with fault-tolerant quantum computing experiments still in the nascent stage, it is difficult to know what constitutes an ideal scheme for local, planar computation. Instead, it is crucial to develop a broad array of tools for fault-tolerance, and hope to meet experiments somewhere in the middle sometime in the future. The triangle code, with its accompanying syndrome extraction circuits and lattice surgery methods, is another tool in this toolkit.

Indeed, in addition to developing planar computation with patches of triangle code, we have also used the notion of a triangle code and a twist to improve surface codes, creating a simple surgery to implement S without state distillation. This is some indication that thinking in terms of triangle codes is worthwhile, even if the triangle code itself is not the resting code of some topologically fault-tolerant architecture. To expedite logical gates, future quantum processors might rather be a tessellation of triangles than a grid.

It would be interesting to generalize the triangle codes to patches with more than one central twist, while still using only weight-4 plaquette stabilizers. One possibility is to take the 3-dimensional visualization of Fig. 2(a) and attach more planes of surface code, creating mixed-type stabilizers and twists where necessary. As one example of this process, we find a topologically closed 2-dimensional surface (conveniently visualized as the surface of a cube with d qubits per edge) containing 8 twists and encoding two logical qubits with distance d . This code is topologically distinct from the toric code. Determining properties or the usefulness of generalized twist-containing patches such as this, and maybe different topologies encoding even more qubits, is a possible direction of future research.

ACKNOWLEDGEMENT

We especially thank Andrew Cross for encouraging us to study this family of codes, for many helpful discussions, and for performing the topological threshold calculation. This work would not have been possible without his guidance. In addition, we thank Sergei Bravyi, Isaac Chuang, and Jay Gambetta for discussions and comments on the manuscript. T.J.Y. acknowledges the hospitality of IBM T.J. Watson where much of this work was done. T.J.Y. also thanks the National Defense Science and Engineering Graduate (NDSEG) fellowship and NSF RQCC Project No. 1111337 for support.

Appendix A: Memory threshold

In this appendix, we first prove that the family of triangle codes possesses a fault-tolerance threshold in the asymptotic limit, and then discuss our simulation for estimating it. The error model is not based on a circuit for syndrome extraction. Single-qubit errors on the data can occur with probability p , and a syndrome bit is flipped with probability q . As is common in these matters, the estimated threshold is much better than the proven one.

Our argument closely follows that of Dennis et. al. [2] who prove a threshold for the toric code. The important differences are (1) our code is not on a closed topological surface and (2) there is a single decoding graph, rather than two separate decoding graphs for X and Z errors (because the triangle code is not CSS). As a consequence of the latter property, while we can prove a threshold with magnitude similar to that for the toric code in the case of uncorrelated X and Z errors, we can prove only a lower threshold (by about a factor of 36) in the case of correlated errors (i.e. single-qubit depolarizing noise).

The decoding graph is overlaid on the triangle code in Fig. 15. To account for errors in measuring the syndrome, we will envision a three-dimensional stack of these decoding graphs, with vertical edges connecting corresponding qubits between the layers. We will call edges connected to only one node “boundary edges” and notice that they can be grouped into three sets corresponding to the side of the triangle code stack they occupy.

Every set of errors E , consisting of any combination of data errors and syndrome errors, can be associated with a subset of edges P in the decoding graph. The association is also true the other way: a subset of edges uniquely specifies a set of errors. Each data qubit has two edges associated with it, a blue edge and a red edge. The blue edge is in P if and only if a Z error occurred on that qubit, and the red edge is in P if and only if an X error did. A Y error corresponds to both edges being in P . Vertical edges between layers correspond to bit flip errors on the syndrome. Using this correspondence we subsequently speak of errors as edges without distinction.

We can also relate logical errors to paths on the decoding graph. A logical error corresponds to a path of length at least d (the code distance) that goes from one boundary edge to another boundary edge occupying a *different* side of the triangle code. The path may traverse several layers before terminating on a different side, though paths that do so are strictly longer than length d .

After T cycles of syndrome extraction, we have accumulated errors E and would like to apply a recovery operation. A simple strategy, and that considered in [2], is to apply a recovery E_m with H_m horizontal edges and V_m vertical edges such that (1) $\partial E = \partial E_m$ and (2) $H_m \log\left(\frac{1-p}{p}\right) + V_m \log\left(\frac{1-q}{q}\right)$ is minimal. Here ∂E for a set of edges E is the boundary of E , the set of nodes connected to only one edge in E . Notice that (2) is defined implicitly for correcting uncorrelated X and Z errors, be-

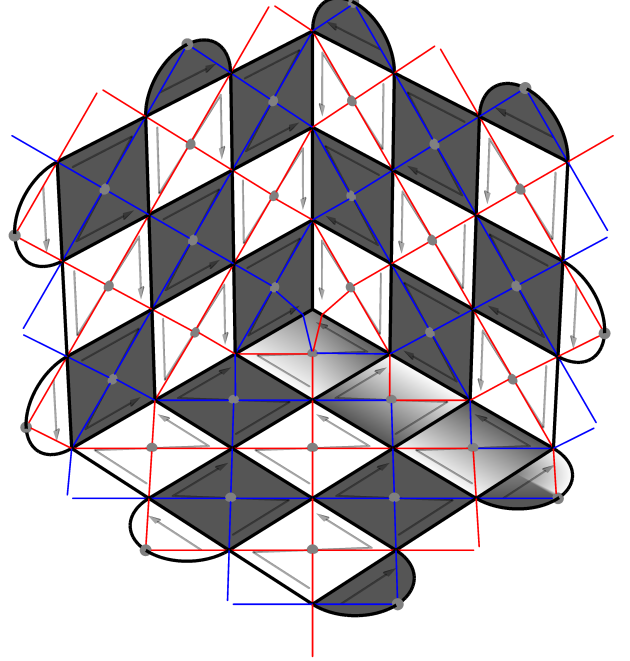


FIG. 15. The decoding graph for a (single layer of) triangle code overlaid on the code. Nodes in the graph are located at each stabilizer (gray dots) and edges are colored blue or red. Some edges are connected to only one node (or alternatively are connected to a node at infinity). These are the boundary edges defined in the text. Also shown in the figure is the order of syndrome extraction as per Fig. 4.

cause it cares about the number of edges in E_m , rather than the number of affected qubits. We will change the recovery later to deal with correlations.

What is the probability that a path P consisting of H horizontal edges and V vertical ones is contained in $E \cup E_m$? If H_e and V_e are the numbers of horizontal and vertical edges in E , then $P \subseteq E \cup E_m$ implies that $H_e + H_m \geq H$ and $V_e + V_m \geq V$. Accordingly,

$$\left(\frac{p}{1-p}\right)^H \left(\frac{q}{1-q}\right)^V \geq \left(\frac{p}{1-p}\right)^{H_m} \left(\frac{q}{1-q}\right)^{V_m} \quad (\text{A1})$$

$$\times \left(\frac{p}{1-p}\right)^{H_e} \left(\frac{q}{1-q}\right)^{V_e}.$$

Using properties (1) and (2) of the recovery, we know that the product of the former two factors on the left side of the inequality is greater than the product of the latter two. This implies

$$\left(\frac{p}{1-p}\right)^{H_e} \left(\frac{q}{1-q}\right)^{V_e} \leq \left(\frac{p}{1-p}\right)^{H/2} \left(\frac{q}{1-q}\right)^{V/2} \quad (\text{A2})$$

The probability E occurs is $\leq p^{H_e}(1-p)^{H-H_e}q^{V_e}(1-q)^{V-V_e}$, and there are $\leq 2^{H+V}$ ways to distribute edges

in P between E and E_m . So, we find the bound

$$\Pr[P \subseteq E \cup E_m] \leq (4p(1-p))^{H/2}(4q(1-q))^{V/2}. \quad (\text{A3})$$

Summing over the possible logical errors bounds the probability of a logical error.

$$\Pr[\text{error}] \leq \sum_{H,V} N_{H,V} (4p(1-p))^{H/2} (4q(1-q))^{V/2}, \quad (\text{A4})$$

where $N_{H,V}$ is the number of paths containing H horizontal and V vertical edges that represent logical errors. For instance, $N_{H,V} = 0$ whenever $H < d$.

While $N_{H,V}$ is difficult to calculate for general $H \geq d$ and V , it is trivial to get a satisfactory bound. Along the boundary of the triangle code stack, which is T layers tall, there are fewer than $T \times (3d)$ boundary edges on which to start a logical error path P . From there, at each node we can typically continue the path in one of at most five directions, three remaining in the same layer and two changing layers. The exception is at the central mixed-type node of any layer, where we have at most six choices. A very rough upper bound is then that there are $N_{H,V} \leq T \times (3d) \times 6^{H+V}$ logical error paths. Putting this together with Eq. (A4), we find that $\Pr[\text{error}]$ vanishes in the limit $d \rightarrow \infty$ whenever $p < p_{\text{th}}$ for some p_{th} that we have now shown is at least 0.7% in the case of uncorrelated X and Z errors.

When X and Z errors are correlated, or, in other words, the data-qubit noise model is single-qubit depolarizing noise, then we should modify property (2) of the recovery to (2') $Q_m \log\left(\frac{1-p}{p}\right) + V_m \log\left(\frac{1-q}{q}\right)$ is minimal. Here Q_m is the number of *qubits* affected by the recovery. This is not the same as the number of horizontal edges, because each qubit is associated with two edges, one blue and one red. However, using the same reasoning as above, we can argue that

$$\Pr[\text{error}] \leq \sum_{Q,V} N'_{Q,V} (4p(1-p))^{Q/2} (4q(1-q))^{V/2}, \quad (\text{A5})$$

where now $N'_{Q,V}$ is the number of paths on the decoding graph containing Q qubits and V vertical edges that also correspond to a logical error. Each qubit is associated with two edges, so $N'_{Q,V} \leq N_{2Q,V} \leq T \times (3d) \times 6^{2Q+V}$. Accordingly, the threshold is $p'_{\text{th}} \geq 0.019\%$ in the case of depolarizing noise.

The corresponding numerical threshold estimates in Table I were calculated in the following way. Storage noise is modeled by applying a channel to each data qubit prior to syndrome measurement. The noise channel is either a bit flip channel with error probability p or the composition of bit flip and phase flip channels, each with error probability p . We do not consider depolarizing noise since error correction by reduction to minimum weight matching is not maximally fault-tolerant for the triangle code for that case (instead, minimum qubit matching as described above Eq. (A5) would be sufficient, but we make no claims about the efficiency of that). As in [2],

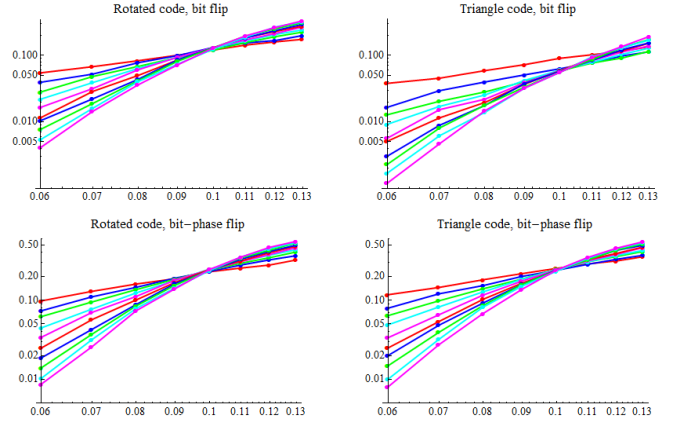


FIG. 16. Logical error rates for bit flip and bit-phase flip noise with ideal syndrome measurements suggest thresholds near 10% for both the triangle and rotated codes. Odd code distances from 3 to 21 are shown, where each point corresponds to either 10,000 or 30,000 Monte-Carlo samples.

we consider two different syndrome measurement models. The “ideal” model applies memory noise, measures one error-free syndrome, and performs error correction. The “noisy” model simulates faulty syndrome measurement using a sequence of $d + 1$ syndrome measurement rounds. In each of the first d rounds, we apply memory noise, measure one error-free syndrome, and flip each syndrome bit with probability p . In the final round, we measure one error-free syndrome. The error correction is computed based on all of the syndrome outcomes.

An algorithm infers error corrections from syndrome outcomes by reduction to minimum weight perfect matching [2]. The edge weights for the matching algorithm are computed differently for the triangle and rotated codes. We use the Manhattan distance for the rotated codes and process bit flip errors and phase flip errors separately. For the triangle code, we construct the decoding graph shown in Fig. 15, assign each edge the same weight, and compute the total weight of any path using Dijkstra’s algorithm. For the “noisy” model, multiple copies of Fig. 15 are joined by edges corresponding to potential syndrome bit errors.

Fig. 16 and Fig. 17 show plots of all of the simulation results. It is worth noting that the error rate of the triangle code is generally much lower than that of the surface code in the presence of only bit flip noise, while it is close to the same in the case of bit-phase flip noise. This is roughly consistent, however, with our counts of lowest-weight logical operators in Tab. V.

Appendix B: Effective distance of syndrome extraction

Our goal in this appendix is to (semi-rigorously) argue that the syndrome extraction circuit in Fig. 4 has effective distance d . That is, there is no set of fewer than d

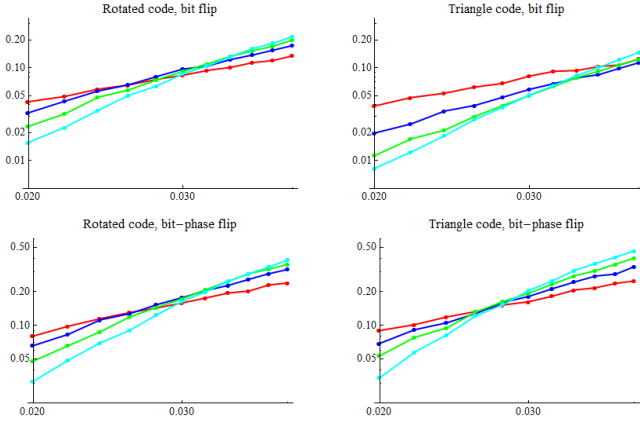


FIG. 17. Logical error rates for bit and bit-phase flip noise with noisy syndrome measurements suggest thresholds near 2.6% for the triangle code and 3.2% for the rotated code. Odd code distances 3, 5, 7, and 9 are shown, where each point corresponds to either 10,000 or 30,000 Monte-Carlo samples.

| | | # of weight d logical strings | | | | |
|----------|--------|---------------------------------|-----|-------|-------|--------|
| code | errors | $d=3$ | 5 | 7 | 9 | 11 |
| surface | X | 8 | 52 | 296 | 1,556 | 7,768 |
| | X,Z | 16 | 104 | 592 | 3,112 | 15,536 |
| | X,Y,Z | 16 | 104 | 592 | 3,112 | 15,536 |
| triangle | X | 3 | 20 | 95 | 546 | 2,583 |
| | X,Z | 14 | 86 | 476 | 2,462 | 12,164 |
| | X,Y,Z | 30 | 204 | 1,164 | 6,072 | 30,012 |

TABLE V. Counting the number of lowest-weight logical operators that can be created from physical errors of various types. While the triangle code has fewer lowest-weight logical operators that can be made from just bit or bit and phase errors, it has many more when correlated errors are allowed.

faults that can cause a logical error. Since Fig. 4 is based on full-distance extraction for the surface code, we first review the argument in that case.

Consider the syndrome extraction circuit indicated by Fig. 18. We aim to show that no set of $< d$ faulty circuit components can cause a logical failure. Because the surface code is CSS, we can consider X and Z errors separately, and refer to a Y error as both an X and a Z . A logical \bar{X} is simply a string of X s between the X -type edges. This implies that, to form \bar{X} , we need at least one X error per column of data qubits. However, we can now check that every faulty circuit component leads to X errors in at most one column. In fact, only “hook” faults (a failure of the second or third gate coupling an ancilla to data) can even leave more than one error on the data. If the syndrome extraction for stabilizer s couples to qubits a, b, c, d in that order, then the errors resulting from hooks take the form $*_c s_d$ and $s_a * _b$, where $*$ is any Pauli and s_j is the j^{th} Pauli of s . Hence, the only hooks that can leave two X errors on

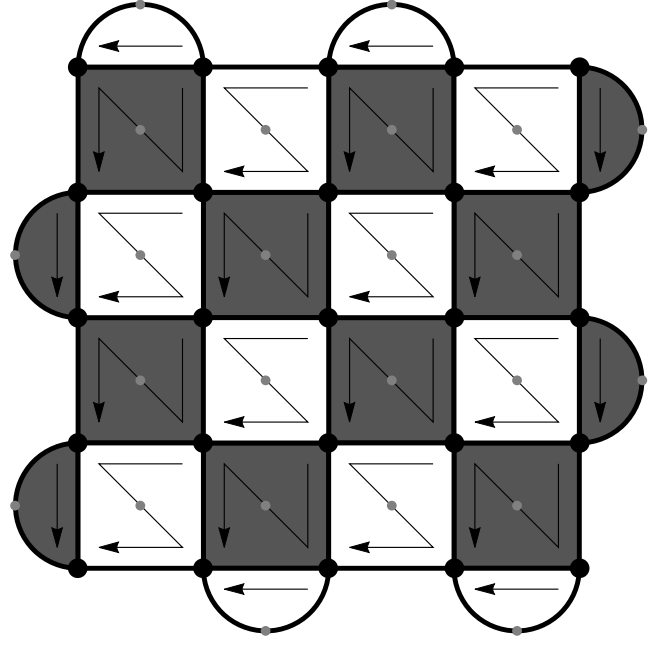


FIG. 18. The pattern for syndrome extraction in the surface code recommended in [5]. Notice that Z -type stabilizers should be measured in a pattern perpendicular to the X -type edges, so that a single hook error cannot spread Z errors to two different rows of data qubits. Likewise, the X -type stabilizers are measured perpendicular to the Z -type edges.

the data are from measuring X -type stabilizers. However, the support of any such hook error lies entirely in one column, because the measurement pattern of the X -type stabilizers is N-shaped. Thus, we have shown that any faulty component only populates at most a single column with X -errors, and therefore at least d are needed to cause \bar{X} . The argument that \bar{Z} requires at least d faults to construct is essentially identical. Constructing \bar{Y} requires constructing both a column traversing string of X s and a row traversing string of Z s. Since X and Z errors are edges on disconnected decoding graphs, \bar{Y} requires at least $2d - 1$ faults.

Now consider syndrome extraction Fig. 4 for the distance- d triangle code. Notice that the planes are simply patches of surface code undergoing syndrome extraction according to Fig. 18, which we know is full-distance. The fact that syndrome measurement is staggered from plane to plane can only make it harder to place an undetected logical error using fewer than d faults (since later measurements may pick up on earlier placed errors). Ignoring the staggered timing, we can still argue for effective distance d of the syndrome extraction.

The outline of the argument is this: we first argue that \bar{Y} cannot be created with fewer than $d - 1$ faults. By symmetry, \bar{X} and \bar{Z} are similarly resilient and this argues that Fig. 4 has effective distance $d - 1$. Then, again appealing to symmetry, we note that the only barrier to achieving effective distance d is the origin. Finally,

case by case analysis of hooks near the origin shows that some hooks are indeed dangerous (two hooks can cause a weight three error as noted in Sec. III of the main text) and that using two 2-CATs for syndrome extraction of checks adjacent to the origin is sufficient to detect the dangerous hooks.

To begin, notice that \bar{Y} must anticommute with any \bar{X} or \bar{Z} . Each row of qubits with fixed x -coordinate $x > 0$ supports a string of X s acting as \bar{X} . Thus, since it must anticommute, \bar{Y} has an odd number of Z s in each of these rows. Define an asymmetric surface code, call it code x , from the qubits in the triangle code with coordinate $x > 0$. Notice that Fig. 4 implies that code x undergoes syndrome extraction with effective distance $(d-1)/2$, because, as described in the case of the surface code Fig. 18, the hooks are correctly oriented. Now, \bar{Y} must act like \bar{Z}_x (i.e. logical Z for code x) when restricted to code x (that is, when the support of \bar{Y} not within code x is ignored). Thus, the full-distance syndrome extraction of code x says that we cannot create \bar{Y} with fewer than $(d-1)/2$ faults.

We can similarly define a surface code from only the qubits at coordinates $z \geq 0$, code z . By its need to anticommute with any \bar{Z} string, \bar{Y} must act like \bar{X}_z (logical X of code z) when restricted to code z . Since code z is undergoing full-distance extraction, it takes at least $(d+1)/2$ faults to make \bar{Y} .

We now combine the two observations that \bar{Y} acts like \bar{Z}_x and \bar{X}_z under the appropriate restrictions. By restricting to codes x and z , we have disconnected the X and Z decoding graphs of Fig. 15. Thus, \bar{Y} restricted to codes x and z cannot be created with fewer than $(d-1)/2 + (d+1)/2 - 1 = d-1$ faults, finishing the argument for effective distance $d-1$.

Now, the location of the mixed-type stabilizers is not fundamental, and local-Clifford equivalent triangle codes exist with it orientated along any ray from the origin. So the argument above could be made with mixed types along any axis instead of the y -axis. What is necessary is that the ray of mixed-type stabilizers terminate in a twist at the origin. Checking hooks at the origin, we see that two hook faults can cause the weight-3 error $Z_{(1,0,0)}Y_{(0,0,0)}X_{(0,0,1)}$, which is three of the supporting Paulis of

$$\bar{Y} = -i \prod_{x \in \mathbb{Z}_s} Z_{(x,0,0)} \prod_{z \in \mathbb{Z}_s} X_{(0,0,z)}, \quad (\text{B1})$$

with $s = (d+1)/2$. This implies \bar{Y} can be made from $d-1$ faults. To catch this case, introduce 2-CAT decoding (Fig. 5) on any two of the central plaquettes, so that any two central hooks can be detected.

Appendix C: exREC pseudothresholds

Here we discuss the fault-tolerant designs and calculations making up Tab. III in the text. All our calculations are rigorous and performed in the exREC formalism

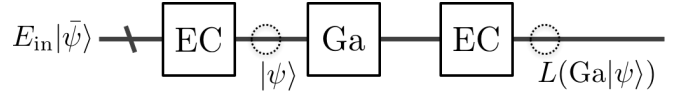


FIG. 19. An exREC [33] consists of a logical gate gadget Ga sandwiched between two error-correction gadgets EC. The gates we consider are transversal, so Ga is simply a depth-1 circuit of single-qubit gates. The EC gadgets are more complex and described in the text. Ideal decoding (dotted circles) after the first and second ECs allows one to determine the Pauli operator L , which depends on faults that have occurred in the exREC, but not, if the CDT criteria [36] are satisfied, on incoming errors E_{in} (here assumed, without loss of generality, to contain no logical errors). If L is the identity, no fault has occurred, but X , Y , or Z signals failure of the exREC. Counting the number of ways failure occurs from ≤ 2 faults gives our pseudothreshold bounds.

[33]. We therefore report pseudothresholds [34] for *computation* (albeit for just Clifford gates on a single logical qubit), as opposed to thresholds for topological memory, which were discussed earlier in Appendix A. Since such pseudothreshold results are not asymptotic, we consider them more suited to comparing small experiments. Eventually one may want to scale up the calculation to simulate the error rates of small algorithms (such as is considered in [35]) made from several exRECs.

In the exREC formalism it is important to create an error-correction (EC) circuit that can recover from any $(d-1)/2$ faults in the preceding gate (Ga) or the preceding EC. In particular, for distance-3, any single fault in the so-called exREC circuit EC.Ga.EC must not lead to a logical error. A logical error in turn is defined as when an ideal decoding of the state after the trailing EC differs from the expected state $\text{Ga}|\psi\rangle$ given that an ideal decoding after the leading EC revealed state $|\psi\rangle$ (see Fig. 19).

All our EC gadgets for distance-3 codes are constructed in the same way. First, define a syndrome extraction (SE) circuit unique to a given design. Then perform SE.SE obtaining two copies of the syndrome. If these syndromes match, the EC gadget is complete, and we attempt to deduce an appropriate recovery (a Pauli operator). If the syndromes do not match, we repeat SE once more, and then attempt recovery. The recovery is never explicitly applied; we simply adjust the Pauli frame [37].

We have to specify how a recovery operator is deduced, often called “decoding” the syndrome. Because we are dealing with finite-sized codes designed specifically for a small demonstration of fault-tolerance, we advocate table lookup for maximum pseudothreshold. The construction of the table proceeds in two steps. First, EC must be able to correct all errors arising from a single fault that it introduces itself, either immediately or, if the error gets through, in the next round of EC. We can enumerate these cases by brute force simulation of the Clifford exREC circuit, and add them to the table. If ever there are two or more errors with the same syndrome but which

commute differently with the logical operators, then we know the SE is not fault-tolerant and must redesign it. The second step of filling the table is to take unused syndromes and assign to each the lowest weight Pauli recovery that corrects the *latest* syndrome measured. Although these syndrome patterns did not arise from single faults, they may arise from double faults, and so we can correct some double faults. However, we make no attempt to correct the most likely (assuming circuit depolarizing noise) double faults consistent with a measured syndrome.

There are two properties of the EC constructed this way that are important. First, it is fault-tolerant in the sense mentioned above that any single fault in the exREC does not cause a logical error. Second, it obeys the Cross-DiVincenzo-Terhal (CDT) criteria [36]. This guarantees that errors incoming to the exREC will not affect the logical failure rate, and thus we can ignore them when calculating pseudothresholds, rather than performing, for instance, an analysis in the context of a worst-case input.

As a detail, we note that SE.SE can sometimes be simplified by further parallelization of the circuit. For instance, Fig. 13 is one such case, where the depth of SE.SE is one less than twice the depth of SE. We perform such simplifications when possible, but do not endeavor to simplify SE.SE.SE if it occurs. This is because in general it cannot be decided whether the third SE should be applied before the second has completely finished.

Having laid out the general procedure for building the EC gadget, we need to now specify SE for each design. We present one SE for each code, and mention the variants, which are straightforward. Afterward, we discuss the counting of malignant faults necessary to actually rigorously bound the pseudothresholds.

Syndrome extraction on the 5-qubit code [38, 39] is typically done with verified 4-CAT states. However, we find verification unsuitable for a small fault-tolerant design, since it requires postselection. In principle, the verification can be removed by decoding the CAT states before measurement [40]. But if we are going to use decoding, there is a more compact strategy using just 3-CATs. In fact, just one set of three ancillas suffices, as shown in Fig. 20, since they can be prepared repeatedly in a 3-CAT state to measure all stabilizers. This is notable as it is the smallest completely fault-tolerant design that we know of, using just 8 qubits including ancillas. Variations on this design using more ancillas are possible, such as using two sets of three ancillas (which is still planar) and four sets of three (which is not), to increase parallelism of the syndrome extraction and correspondingly the threshold.

For the smallest color code, equivalent to Steane's code [26], it has been suggested that 2-CATs are sufficient for full-distance syndrome extraction [16, 41]. Indeed we find this to be the case, *if* we decode the 2-CATs to detect hook errors. In fact, if we did not decode, the high symmetry of the 7-qubit code renders any undetected hook error fatal. The smallest fault-tolerant planar design is

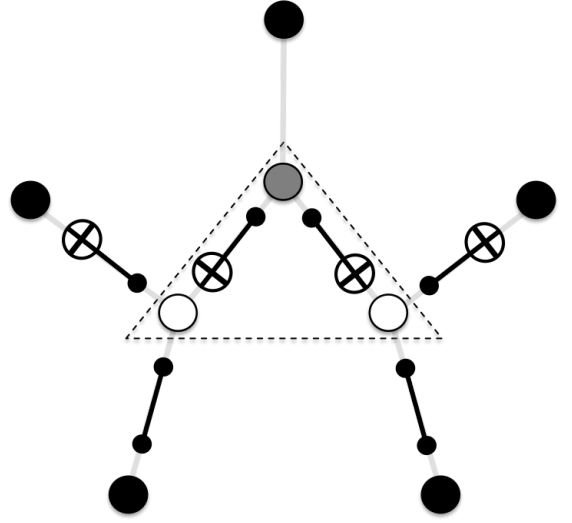


FIG. 20. A layout of the 5-qubit code (black circles) and ancilla qubits (gray, prepared as $|+\rangle$, and white, prepared as $|0\rangle$) overlaid with the circuit for measurement of a single stabilizer $XZZXI$. The gates inside the dotted triangle are performed twice, both before and after the gates outside the triangle. For fault-tolerance the order of all gates is otherwise irrelevant, though it will change the decoding table. All ancilla qubits are measured afterward in the same basis they were prepared. Light lines in the background indicate the required qubit interaction graph for an entire SE cycle, which is planar and only degree three.

shown in Fig. 21 where each face possesses two ancillas. These must be prepared and repared in 2-CAT states to measure all stabilizers (since each face represents both an X -type and Z -type stabilizer). If instead we introduce four ancillas per face such that the X - and Z -syndromes might be extracted more in parallel (e.g. the ancillas for Z -type can now be prepared while those for X -type are being measured), we lose planarity but increase the threshold.

The Bacon-Shor code is unique in our study as it is a subsystem code. When syndrome extraction is performed with ancilla codeblocks, its threshold is one of the highest known [42]. But as pointed out in [42], we can exploit the subsystem structure to measure only weight-two operators, also called gauge operators, to deduce the syndrome. Since weight-two measurements cannot introduce more than weight-one errors to the data, this procedure is naturally fault-tolerant. To minimize ancilla reuse, while keeping under 20 total qubits (our cutoff for “small” fault-tolerant designs), we use a total of eight ancillas, Fig. 22. SE consists of measuring all X -type gauge operators once, followed by measurement of all the Z -type gauge operators.

We have described several SE designs for the triangle code in Section V. The three in Tab. III are (1) the two 2-CAT design suggested by the distance-3 version of Fig. 4, (2) the degree-5 version of Fig. 13 wherein plaquette ancillas can interact directly with neighboring loop ancillas,

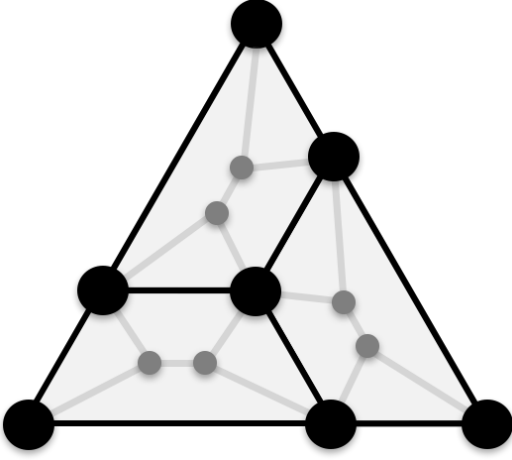


FIG. 21. A 6-ancilla design for the smallest color code. With only degree-3 connectivity (light-gray edges), this layout provides fully fault-tolerant syndrome extraction by creating and decoding 2-CATs.

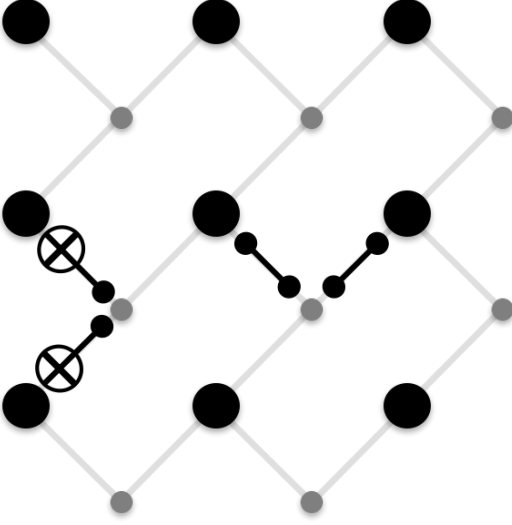


FIG. 22. The 3×3 Bacon-Shor code with 8 ancilla qubits. Overlaid are the coupling gates required to measure an X-type gauge operator (to the left) and a Z-type gauge operator (right) when the ancillas are prepared and measured in the X-basis. Light-gray lines indicate the degree-3 connectivity required for full syndrome extraction.

and (3) the degree-4 version of Fig. 13 wherein the ancillas interact only via a data qubit intermediary.

The surface code SE is taken from [5]. There they conclude that the 8-ancilla version possesses the highest threshold. We also consider the 4-ancilla version, in which plaquette ancillas are reused to measure loops, because it uses 13 total qubits and so offers an equal-qubit comparison to our smallest triangle code design.

Once we have completely specified the exREC for a design, we can rigorously bound the logical error rate. Our error model is standard. Single-qubit operations includ-

ing identity gates fail with X , Y , or Z with probability $p/3$ while two-qubit gates fail in one of 15 Pauli ways with probability $p/15$ each. Single-qubit preparation of $|0\rangle$ or $|+\rangle$ fails (by a Pauli error X or Z , respectively) with probability p . Measurement in either the X - or Z -basis fail (by reporting the wrong bit) with probability p . All circuit components succeed or fail independently.

Counting malignant sets proceeds similarly to [33], though our bounds are calculated using different (slightly tighter) formulas. We can calculate the quantities

$$P_{\text{fail}}^{(2)} = \Pr[L \neq I, \leq 2 \text{ faults}], \quad (\text{C1})$$

$$P_{\text{succ}}^{(2)} = \Pr[L = I, \leq 2 \text{ faults}] \quad (\text{C2})$$

(where I is the 2×2 identity) by enumerating all sets of at most two faults and determining L (see Fig. 19) for each. The probability that a particular set of faults occurs is the product of the probabilities for the faulty components failing and for all other components succeeding.

Defining $P_{\text{fail}} = \Pr[L \neq I]$, we then see the bounds,

$$P_{\text{fail}}^{(2)} \leq P_{\text{fail}} \leq 1 - P_{\text{succ}}^{(2)}. \quad (\text{C3})$$

By the fault-tolerance of our designs, $P_{\text{fail}} = O(p^2)$, and likewise with the upper and lower bounds. So the pseudthreshold p_{th} found by solving $P_{\text{fail}}(p_{\text{th}}) = p_{\text{th}}$ is bounded by the solutions to

$$P_{\text{fail}}^{(2)}(p_{\text{upp}}) = p_{\text{upp}}, \quad (\text{C4})$$

$$1 - P_{\text{succ}}^{(2)}(p_{\text{low}}) = p_{\text{low}} \quad (\text{C5})$$

like $p_{\text{low}} \leq p_{\text{th}} \leq p_{\text{upp}}$.

Note that counting sets of faults is complicated by the fact our EC is non-deterministic (the third application of SE is conditional on the first two not matching). Thus, to calculate $P_{\text{fail}}^{(2)}$ and $P_{\text{succ}}^{(2)}$ we must break each into four cases. This amounts to the identities

$$P_{\text{fail}}^{(2)} = \sum_{i,j \in \{2,3\}} \Pr[L \neq I, \leq 2 \text{ faults}, i\text{-EC}_1, j\text{-EC}_2] \quad (\text{C6})$$

$$P_{\text{succ}}^{(2)} = \sum_{i,j \in \{2,3\}} \Pr[L = I, \leq 2 \text{ faults}, i\text{-EC}_1, j\text{-EC}_2], \quad (\text{C7})$$

where $i\text{-EC}_1$ indicates that the first EC consists of two ($i = 2$) or three ($i = 3$) SE, and likewise with $j\text{-EC}_2$ for the second EC. The probabilities in the sums can be calculated by simulating all sets of at most two faults in a circuit with an $i\text{-EC}_1$ and a $j\text{-EC}_2$ and then keeping only the sets of faults which cause syndromes that are consistent with the presence of an $i\text{-EC}_1$ and a $j\text{-EC}_2$.

Having discussed gate exRECs, we briefly describe the procedure for calculating the error rate of measurement exRECs, which we performed for the small triangle code designs in Tab. IV. A measurement exREC is set up similarly to Fig. 19, except that the second EC is replaced by a measurement EC, or MEC. Thus, the measurement exREC includes both the final single-qubit gate in the circuit and the measurement.

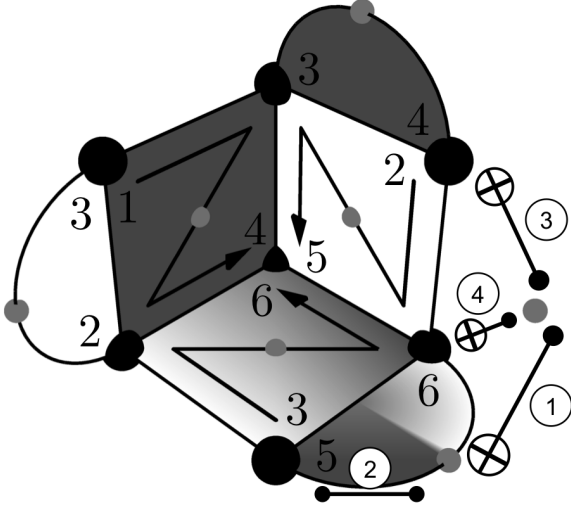


FIG. 23. Measuring \bar{Y} using one extra qubit beyond Fig. 13. Gates are labeled with the timesteps in which they should be applied. The extra qubit is prepared (during timestep zero) in $|+\rangle$ and the loop ancilla for the mixed-type loop is prepared in $|0\rangle$ so that the first CNOT creates a 2-CAT state for measuring \bar{Y} . This allows the loop ancilla to be measured and reprepared in $|+\rangle$ to measure the loop stabilizer in accordance with the scheduling of Fig. 13. Thus, measuring \bar{Y} and all the stabilizers takes no more timesteps than measuring just the stabilizers.

The MEC also consists of repeated syndrome extraction, but one that measures the eigenvalue of a logical Pauli \bar{P} in addition to the syndromes of all stabilizers. Call this extraction an MSE. Then, an MEC consists of two repeats of the MSE, which is followed by another MSE if the first two did not match. The decoding table of the MSE is constructed just as before for the SE, but, because the initial logical state is unknown, the \bar{P} values measured cannot provide information about the error directly — only when they change across different MSEs is it relevant.

The MSE can in principle be done in many different ways. For instance, the procedure described in Section V reuses the plaquette ancilla in the xy -plane to do the $\bar{P} = \bar{Z}$ measurement. In Fig. 23, we show how a measurement with one extra ancilla qubit can be done. The counting of malignant sets also proceeds similarly as the gate case, except that final logical errors of \bar{P} are not considered failures.

-
- [1] Sergey B Bravyi and A Yu Kitaev, “Quantum codes on a lattice with boundary,” quant-ph/9811052 (1998).
 - [2] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill, “Topological quantum memory,” *Journal of Mathematical Physics* **43**, 4452–4505 (2002).
 - [3] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland, “Surface codes: Towards practical large-scale quantum computation,” *Physical Review A* **86**, 032324 (2012).
 - [4] Dorit Aharonov and Lior Eldar, “On the complexity of commuting local Hamiltonians, and tight conditions for topological order in such systems,” in *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on* (IEEE, 2011) pp. 334–343.
 - [5] Yu Tomita and Krysta M Svore, “Low-distance surface codes under realistic quantum noise,” *Physical Review A* **90**, 062320 (2014).
 - [6] David S Wang, Austin G Fowler, and Lloyd CL Hollenberg, “Surface code quantum computing with error rates over 1%,” *Physical Review A* **83**, 020302 (2011).
 - [7] Adrian Hutter, James R Wootton, and Daniel Loss, “Efficient Markov chain Monte Carlo algorithm for the surface code,” *Physical Review A* **89**, 022326 (2014).
 - [8] Sergey Bravyi, Martin Suchara, and Alexander Vargo, “Efficient algorithms for maximum likelihood decoding in the surface code,” *Physical Review A* **90**, 032326 (2014).
 - [9] James R Wootton and Daniel Loss, “High threshold error correction for the surface code,” *Physical Review Letters* **109**, 160503 (2012).
 - [10] Austin G Fowler and Simon J Devitt, “A bridge to lower overhead quantum computation,” arXiv:1209.0510 (2012).
 - [11] Héctor Bombín, “Topological subsystem codes,” *Physical Review A* **81**, 032301 (2010).
 - [12] Sergey Bravyi, Guillaume Duclos-Cianci, David Poulin, and Martin Suchara, “Subsystem surface codes with three-qubit check operators,” arXiv:1207.1443 (2012).
 - [13] Héctor Bombín and Miguel Angel Martin-Delgado, “Topological quantum distillation,” *Physical Review Letters* **97**, 180501 (2006).
 - [14] Héctor Bombín, “Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes,” *New Journal of Physics* **17**, 083002 (2015).
 - [15] Aleksander Kubica and Michael E Beverland, “Universal transversal gates with color codes: A simplified approach,” *Physical Review A* **91**, 032330 (2015).
 - [16] Andrew J Landahl and Ciaran Ryan-Anderson, “Quantum computing by color-code lattice surgery,” arXiv:1407.5103 (2014).
 - [17] Robert Raussendorf and Jim Harrington, “Fault-tolerant quantum computation with high threshold in two dimensions,” *Physical Review Letters* **98**, 190504 (2007).
 - [18] Sergey Bravyi and Alexei Kitaev, “Universal quantum computation with ideal clifford gates and noisy ancillas,” *Physical Review A* **71**, 022316 (2005).
 - [19] Austin G Fowler, Simon J Devitt, and Cody Jones, “Surface code implementation of block code state distillation,” *Scientific reports* **3** (2013).

- [20] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski, “High-threshold universal quantum computation on the surface code,” *Physical Review A* **80**, 052312 (2009).
- [21] Austin G Fowler, “Time-optimal quantum computation,” arXiv:1210.4626 (2012).
- [22] Clare Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter, “Surface code quantum computing by lattice surgery,” *New Journal of Physics* **14**, 123011 (2012).
- [23] Héctor Bombín, “Topological order with a twist: Ising anyons from an Abelian model,” *Physical Review Letters* **105**, 030403 (2010).
- [24] Matthew B Hastings and A Geller, “Reduced space-time and time costs using dislocation codes and arbitrary ancillas,” arXiv:1408.3379 (2014).
- [25] A. R. Calderbank and Peter W. Shor, “Good quantum error-correcting codes exist,” *Phys. Rev. A* **54**, 1098–1105 (1996).
- [26] Andrew M Steane, “Error correcting codes in quantum theory,” *Phys. Rev. Lett.* **77**, 793 (1996).
- [27] Benjamin J Brown, Katharina Laubscher, Markus S Kesselring, and James R Wootton, “Poking holes and cutting corners to achieve Clifford gates with the surface code,” arXiv:1609.04673 (2016).
- [28] Daniel Gottesman and Isaac L Chuang, “Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations,” *Nature* **402**, 390–393 (1999).
- [29] Peter Brooks and John Preskill, “Fault-tolerant quantum computation with asymmetric Bacon-Shor codes,” *Physical Review A* **87**, 032310 (2013).
- [30] Jonathan E Moussa, “Transversal clifford gates on folded surface codes,” arXiv preprint arXiv:1603.02286 (2016).
- [31] Aleksander Kubica, Beni Yoshida, and Fernando Pastawski, “Unfolding the color code,” *New Journal of Physics* **17**, 083026 (2015).
- [32] Andrew W. Cross, personal communication.
- [33] Panos Aliferis, Daniel Gottesman, and John Preskill, “Quantum accuracy threshold for concatenated distance-3 codes,” *Quantum Information & Computation* **6**, 97–165 (2006).
- [34] Krysta M Svore, Andrew W Cross, Isaac L Chuang, and Alfred V Aho, “A flow-map model for analyzing pseudothresholds in fault-tolerant quantum computing,” *Quantum Information & Computation* **6**, 193–212 (2006).
- [35] Daniel Gottesman, “Quantum fault tolerance in small experiments,” arXiv preprint arXiv:1610.03507 (2016).
- [36] Andrew W Cross, David P Divincenzo, and Barbara M Terhal, “A comparative code study for quantum fault-tolerance,” arXiv:0711.1556 (2007).
- [37] Emanuel Knill, “Quantum computing with realistically noisy devices,” *Nature* **434**, 39–44 (2005).
- [38] Charles H Bennett, David P DiVincenzo, John A Smolin, and William K Wootters, “Mixed-state entanglement and quantum error correction,” *Physical Review A* **54**, 3824 (1996).
- [39] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek, “Perfect quantum error correcting code,” *Physical Review Letters* **77**, 198 (1996).
- [40] David P DiVincenzo and Panos Aliferis, “Effective fault-tolerant quantum computation with slow measurements,” *Physical Review Letters* **98**, 020501 (2007).
- [41] Ashley M Stephens, “Efficient fault-tolerant decoding of topological color codes,” arXiv preprint arXiv:1402.3037 (2014).
- [42] Panos Aliferis and Andrew W Cross, “Subsystem fault tolerance with the Bacon-Shor code,” *Physical Review Letters* **98**, 220502 (2007).